Fabien DONIUS, Nicolas GRILL, Chérine KAMEL, Selim MILED - Ing1 Gr4

BILAN / RECETTE

Les codes correcteurs d'erreur

Préface

Dans ce projet scientifique et technique en équipe sur les codes correcteurs d'erreurs, présenté par le groupe d'étudiants compose de : Fabien DONIUS, Nicolas GRILL, Chérine KAMEL et Selim MILED, on trouve une bonne introduction sur le thème et ces applications sur des thèmes réels.

Le groupe d'étudiants a présenté un bon travail du point de vue de la programmation sur le fait que leur code corrige correctement les erreurs aperçu ou détecté sur des fichiers de données, ainsi que sur du point de vue de la rédaction et de synthèse. Au cours de leur projet, ils ont montré leur adaptation pour aborder des thèmes de recherches qui demande un travail personnel au delà de l'état d'avancement du programme étudié au niveau de l'école en première année ECE.

Je suis persuade que ce projet (le programme et le rapport) constitue un outil intéressant pour tous ceux qui veulent en savoir plus sur les correcteurs d'erreurs et leurs applications. Pour terminer j'insiste sur la qualité du travail accompli par les étudiants dans leur projet.

Palaiseau, le 05/04/2007

Houari MECHKOUR

Bilan et recette

Sommaire

PRÉFACE 2

SOMMAIRE 3

RÉSULTATS DU PROJET 4

I. ÉTUDES PRÉLIMINAIRES 4
II. ANALYSE THÉORIQUE : CODE DE HAMMING(7,4) 4
III. ANALYSE PRATIQUE : SIMULATION SOUS CCE TESTEUR 5

BILAN DU PROJET, RÉPONSE AUX ATTENTES 5

LISTE DES ANNEXES 6

D'après le second théorème de Shannon, si un canal de transmission génère des erreurs, il est toujours possible de récupérer le message d'origine à condition de lui ajouter suffisamment de redondance.

L'application de ce théorème est incarnée par les codes correcteurs d'erreurs. Codes cycliques, matriciels, convolutifs ou turbo codes, autant de directions différentes à emprunter pour comprendre ce vaste sujet peu connu, et pourtant à la base du développement des technologies de l'information et de la communication.

L'ensemble de notre projet avait pour but de comprendre ces codes dans un premier temps, puis de chercher à dresser des conclusions sur leurs usages et leurs performances. Ce dossier présente les résultats obtenus, puis dresse un bilan par rapport aux attentes posées par notre cahier des charges.

Résultats du projet

Comme prévu, notre travail a consisté dans un premier temps en une analyse théorique des codes correcteurs d'erreurs. Grâce à ces résultats, nous avons pu poser les bases nécessaires à la création d'un outil de comparaison performant, notre logiciel : CCE testeur.

I. Études préliminaires

Nous avons commencé par répertorier les différents types de codes, ainsi que les bases de leur fonctionnement. Ceci a permis d'orienter notre travail théorique selon des critères de faisabilité et d'intérêt.

➔ Cyclic Redundancy Codes (CRC)

Ils consistent à transformer une série de bits en un polynôme de degré n (longueur du paquet) (typiquement 8, pour un octet) par multiplication avec un polynôme dit générateur. A la réception, une division Euclidienne du message reçu par le polynôme générateur permet de savoir s'il y a eu des erreurs de transmission, auquel cas le reste sera non-nul. S'ils sont très performants, voire infaillibles pour certains, dans le domaine de la détection d'erreur, leur capacité de correction est nulle.

➔ Codes convolutifs

Les codes convolutifs sont les plus utilisés aujourd'hui dans les télécommunications fixes et mobiles, grâce à leur souplesse et leur efficacité. Ils ont été introduits en 1955 par Elias et sont une branche des codes en blocs linéaires apportant ainsi des propriétés favorables au codage et à l'amélioration des performances

➔ Turbo codes

Certainement les codes correcteurs les plus puissants à ce jour. Cependant si leurs performances ne font aucun doute, personne à ce jour n'a réussi à démontrer leur fonctionnement. Il est alors évident que nous ne pouvions nous y intéresser.

➔ Codes matriciels

Basés comme leur nom l'indique sur des matrices, ces codes offrent des performances très raisonnables par rapport aux besoins actuels. Si la très grande majorité demande un niveau élevé en mathématiques et théorie de l'information pour être étudiés, certains sont plus abordables. Notre choix était fait.

II. Analyse théorique : code de Hamming(7,4)

La démonstration détaillée de ce code se trouve dans l'annexe au dossier.

Elle nous a permis dans un premier temps de généraliser le fonctionnement d'un code correcteur, la manière dont il encode et traite les données, comment sont appliquées les corrections, en bref tout le processus suivi.

En poussant l'étude, nous avons pu déterminer jusqu'où ce code fonctionne correctement. En effet au-delà d'un certain pourcentage d'erreurs dans le message ou lorsqu'il y a plus d'une erreur dans un même paquet, la correction n'est pas totale. Plus étonnant encore, dans certains cas Hamming(7,4) créé des erreurs supplémentaires.

Ces conclusions sont très importantes pour notre projet, car elles résument tout ce qui peut affecter le fonctionnement d'un code correcteur : la successivité des erreurs et leur taux trop élevé. Il nous fallait donc prendre cela en compte dans la conception du logiciel.

III. Analyse pratique : simulation sous CCE testeur

Notre logiciel, CCE testeur (dont le fonctionnement et l'architecture sont détaillés en annexe), permet de tester les codes correcteurs d'erreurs et ainsi les comparer entre eux dans différentes situations, rapidement et précisément.

La simulation effectuée permet de comparer les codes de Hamming(7,4) et Golay(24,12). Elle sert également de simulation-type pour toute comparaison de code qui pourra être faite ultérieurement.

Voir l'annexe *Comparatif Hamming(7,4) Golay(24,12)*

Bilan du projet, réponse aux attentes

Dans notre cahier des charges, nous établissions comme but à ce projet l'établissement d'une hiérarchie des codes correcteurs d'erreurs selon leur usage optimal dans les différents cas représentatifs des conditions réelles courantes.

Nos conclusions sur ce point sont les suivantes :

➔ Forts de leur capacité de détection quasiment infaillible, les Codes de Redondance Cycliques (codes détecteurs d'erreurs) restent la meilleure option pour peu que l'on dispose d'une bande passante suffisamment large pour retransmettre les données, d'une vitesse de transmission importante et qu'un léger décalage temporel dans la réception de certains paquets ne soit pas un problème. Dans le cas par exemple d'une diffusion multimédia en simultané, type Télévision Numérique Terrestre, cette solution est inenvisageable.

➔ Les performances des codes correcteurs matriciels dépendent du taux de redondance qu'ils intègrent. Si à taux et occurrences d'erreurs faibles ces codes s'avèrent être avantageux (notamment dans les cas ou leur temps d'exécution est inférieur au temps nécessaire à la retransmission d'un paquet), dans des conditions plus extrêmes la redondance nécessaire devient telle que le message transmis est beaucoup plus volumineux que le message d'origine. Ils restent donc réservés à des canaux de transmission à très large capacité mais disposant de vitesses très basses. Le cas le plus typique est celui de la communication avec des entités spatiales : sondes, navettes, satellites…

➔ Les Turbo codes sont de très loin les codes correcteurs les plus performants. Nous l'avons appris de plusieurs sources, mais nous sommes, comme toute la communauté scientifique, dans l'incapacité totale de les comprendre. Cela n'empêche pas leur utilisation, par exemple pour l'UMTS ou encore l'ADSL 2.

Le cas des Turbo codes est révélateur du plus grand problème que nous avons rencontré au cours de ces mois de travail : notre incapacité à expliquer ce que nous étudiions. Et c'est là la cause du décalage entre nos objectifs de début de projet et notre rendu final. Ne connaissant pas du tout le domaine, nous avons tout d'abord sous-estimé sa complexité, supposant qu'il était « forcément à la hauteur d'Ing1 ». Et par conséquent, nous pensions pouvoir faire le tour du sujet,

voire même plus. Quelques semaines plus tard, nous nous rendions compte que nous nous attaquions à un domaine scientifique à part entière, et non une sous-branche infime des mathématiques ou de l'informatique.

Ainsi, si l'on interprète notre cahier des charges par : « à partir de l'analyse mathématique de la totalité des codes correcteurs d'erreurs, nous dresserons un tableau comparatif exhaustif selon tous les paramètres possibles », nous n'avons à présenter qu'un échec cuisant, que même les spécialistes du domaine n'auraient su éviter.

En revanche, si l'on interprète nos objectifs par : « comprendre les paramètres intervenant dans la génération puis la correction des erreurs dans la transmission d'informations, et ainsi créer un modèle rendant des simulations précises possibles », on peut certainement dire que notre projet est un succès.

Grâce à l'analyse théorique de cas particuliers à la limite de nos compétences, nous avons su tirer des généralisations qui ont été les bases de CCE testeur. De par son évolutivité et sa modularité, notre outil est la matérialisation de ce que l'on aurait aimé pouvoir faire de nous même : prendre n'importe quel code correcteur d'erreur, et pouvoir analyser ses différentes réactions dans une infinité de contextes reposant sur des variables réalistes. Plus qu'une modélisation, ce logiciel est en lui-même le début de réponse aux questions que nous nous étions posées dès le départ.

L'autre point important au sujet de CCE testeur est qu'il synthétise tout ce que nous avons appris au cours de ces plusieurs mois de recherche et d'analyse. Chaque ligne de code relue nous ramène à ce jour où nous nous sommes demandé ce qu'était un code correcteur. Si le but premier des PSTE était de faire de nous des amateurs experts en ce domaine, le bilan est très clairement positif.

Enfin, sur le plan du travail en équipe, nous avons également beaucoup gagné. Apprendre à répartir autant de tâches équitablement n'a pas toujours été facile, et il est arrivé que certains d'entre nous se retrouvent confrontés à une charge de travail plus importante que d'autres. Ce phénomène associé aux tensions des dates butoirs et aux échecs parfois inévitables a engendré certaines situations de crises au sein de la Team CCE. L'essentiel reste que nous avons toujours su dépasser ces moments de tensions, aller au-delà des obstacles ou parvenir à les détourner, le tout au service de notre objectif commun. Aux vues du résultat, on peut donc estimer que notre organisation était satisfaisante, mais cependant perfectible.

Liste des annexes

Analyse mathématique du code de Hamming(7,4)

Analyse mathématique du code de Golay(24,12)

Comparaison de deux codes correcteurs : Golay(24,12) et Hamming(7,4)

Documentation du logiciel CCE testeur avec fichier d'aide

Comment créer sa dll de code correcteur ou de génération d'erreur à implémenter au logiciel et un fichier ressource permettant la personnalisation du résultat en HTML.

L'ensemble de ces documents et de tous ceux rendus au cours de l'année, le logiciel et tout autre fichier en rapport avec le projet sont disponibles sur le CD ci-joint et à l'adresse suivante : http://cce-project.crae-prod.info/.